REPORT - Address Abyss

Problem Statement :

"A massive dump of network logs hides a secret. Only entries with IPs in the ranges 92.7.X.Y (IPv4) and 2510:a1:X::Y (IPv6-like) matter. Each X is an index, each Y a value. Extract the pairs, arrange the values by their indices, and reconstruct the flag from the fragments. We have already tried doing this. Here's a snippet of our analysis:

Valid IP: 92.7.2.0 -> index=2, char=0

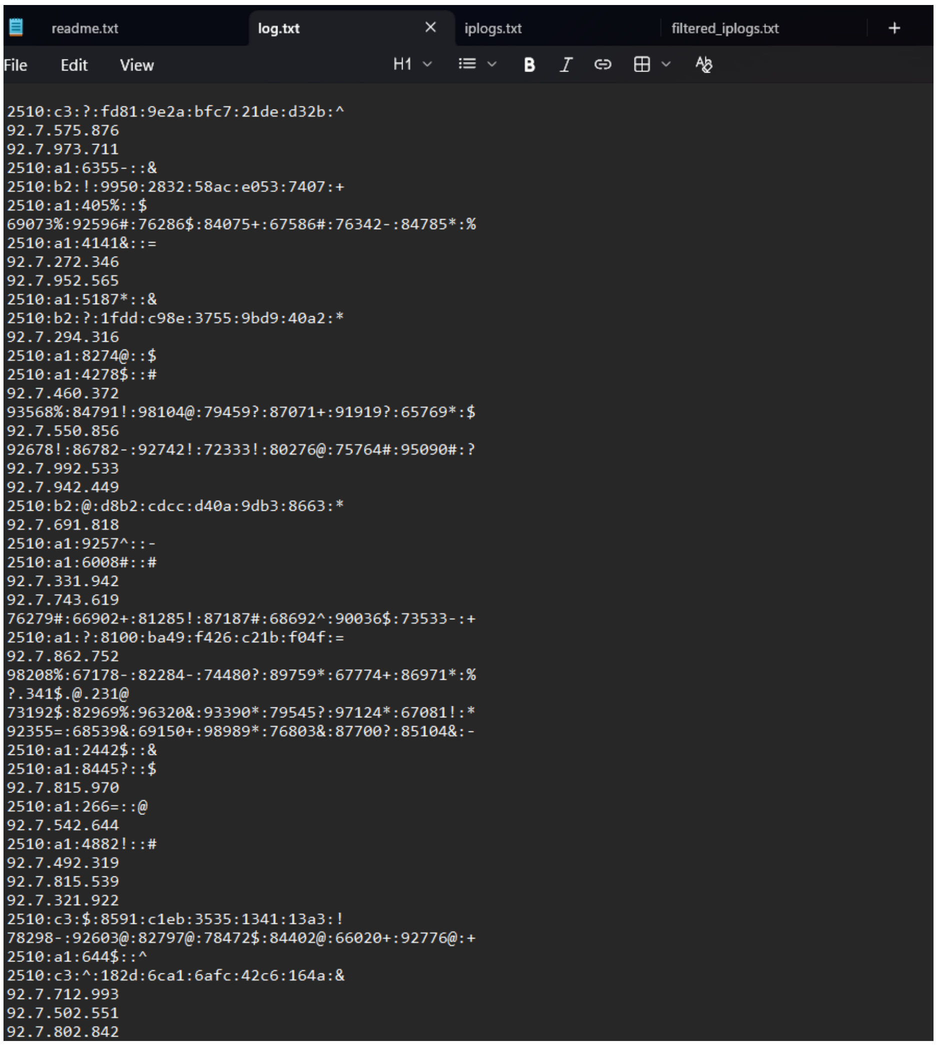
Valid IP: 2510:a1:5::e -> index=5, char=e

Valid IP: 2510:a1:b::f -> index=11, char=f

Once you find all the missing pieces build the flag in format HQX{..0..e.....f....}"

METHODOLOGY

We start by unzipping the file to find a txt for the ip log file with a lot of data (noise data mainly) { log.txt }

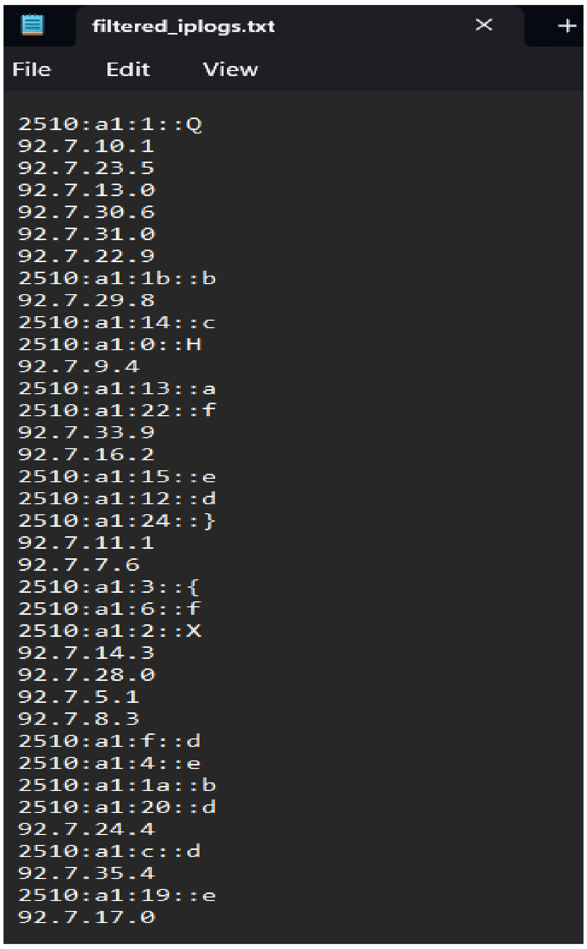


```
2510:c3:?:fd81:9e2a:bfc7:21de:d32b:^
92.7.575.876
92.7.973.711
2510:a1:6355-::&
2510:b2:!:9950:2832:58ac:e053:7407:+
2510:a1:405%::$
69073%:92596#:76286$:84075+:67586#:76342-:84785*:%
2510:a1:4141&::=
92.7.272.346
92.7.952.565
2510:a1:5187*::&
2510:b2:?:1fdd:c98e:3755:9bd9:40a2:*
92.7.294.316
2510:a1:8274@::$
2510:a1:4278$::#
92.7.460.372
93568%:84791!:98104@:79459?:87071+:91919?:65769*:$
92.7.550.856
92678!:86782-:92742!:72333!:80276@:75764#:95090#:?
92.7.992.533
92.7.942.449
2510:b2:@:d8b2:cdcc:d40a:9db3:8663:*
92.7.691.818
2510:a1:9257^::-
2510:a1:6008#::#
92.7.331.942
92.7.743.619
76279#:66902+:81285!:87187#:68692^:90036$:73533-:+
2510:a1:?:8100:ba49:f426:c21b:f04f:=
92.7.862.752
98208%:67178-:82284-:74480?:89759*:67774+:86971*:%
?.341$.@.231@
73192$:82969%:96320&:93390*:79545?:97124*:67081!:*
92355=:68539&:69150+:98989*:76803&:87700?:85104&:-
2510:a1:2442$::&
2510:a1:8445?::$
92.7.815.970
2510:a1:266=::@
92.7.542.644
2510:a1:4882!::#
92.7.492.319
92.7.815.539
92.7.321.922
2510:c3:$:8591:c1eb:3535:1341:13a3:!
78298-:92603@:82797@:78472$:84402@:66020+:92776@:+
2510:a1:644$::^
2510:c3:^:182d:6ca1:6afc:42c6:164a:&
92.7.712.993
92.7.502.551
92.7.802.842
```

- So we start by firstly filtering the valid ip addresses.. we use this python script to do the same

```python
# Define the set of special characters to remove
special_chars = set("!@$#%^&*()[]<>?/\\|`~-+=")


def is_valid_ipv4(ip):
   parts = ip.split('.')
   if len(parts) != 4:
       return False
   try:
       for p in parts:
           num = int(p)
           if not (0 <= num <= 255):
               return False
       return True
   except ValueError:
       return False


with open("iplogs.txt", "r") as infile, open("filtered_iplogs.txt", "w") as outfile:
   for line in infile:
       line = line.strip()


       if line.startswith("92.7") or line.startswith("2510:a1"):
           # Remove lines with special characters
           if any(char in special_chars for char in line):
               continue


           if line.startswith("92.7"):
               ip_part = line.split()[0]
               if not is_valid_ipv4(ip_part):
                   continue   # skip invalid IPv4


           # Otherwise keep line
           outfile.write(line + "\n")


print("Done. Check filtered_iplogs.txt")
```

- Now we generated filtered_iplogs.txt file with the filtered ip addresses.



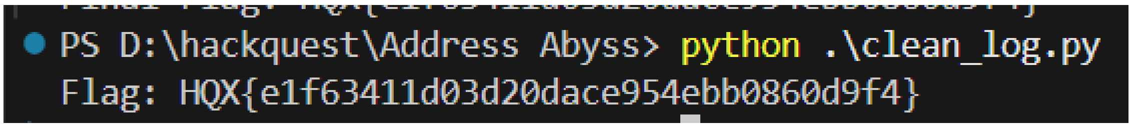
```
2510:a1:1::Q
92.7.10.1
92.7.23.5
92.7.13.0
92.7.30.6
92.7.31.0
92.7.22.9
2510:a1:1b::b
92.7.29.8
2510:a1:14::c
2510:a1:0::H
92.7.9.4
2510:a1:13::a
2510:a1:22::f
92.7.33.9
92.7.16.2
2510:a1:15::e
2510:a1:12::d
2510:a1:24::}
92.7.11.1
92.7.7.6
2510:a1:3::{
2510:a1:6::f
2510:a1:2::X
92.7.14.3
92.7.28.0
92.7.5.1
92.7.8.3
2510:a1:f::d
2510:a1:4::e
2510:a1:1a::b
2510:a1:20::d
92.7.24.4
2510:a1:c::d
92.7.35.4
2510:a1:19::e
92.7.17.0
```

- We can see now just the required 37 lines here which will be used to make our flag
- # Format: 92.7.[index].[char]
- # Format: 2510:a1:[hex_idx]::[char]
- # We split by '::' to get the char, and then look at the prefix for the hex

For this we run another scripts that does the work for us

```python
def solve_abyss_final(data):
    pieces = {}

    for line in data.strip().split('\n'):
        line = line.strip()
        if not line: continue

        if line.startswith("92.7"):
            # Format: 92.7.[index].[char]
            parts = line.split('.')
            idx = int(parts[2])
            char = parts[3]
            pieces[idx] = char

        elif line.startswith("2510:a1"):
            # Format: 2510:a1:[hex_idx]::[char]
            # We split by '::' to get the char, and then look at the prefix for the hex
            prefix, char = line.split('::')
            hex_val = prefix.split(':')[-1]
            idx = int(hex_val, 16)
            pieces[idx] = char
    # Sort indices and build string
    max_idx = max(pieces.keys())
    flag = "".join(pieces.get(i, "?") for i in range(max_idx + 1))
    return flag

logs = """
2510:a1:1::Q
92.7.10.1
92.7.23.5
92.7.13.0
92.7.30.6
92.7.31.0
92.7.22.9
2510:a1:1b::b
92.7.29.8
2510:a1:14::c
2510:a1:0::H
92.7.9.4
2510:a1:13::a
2510:a1:22::f
92.7.33.9
92.7.16.2
2510:a1:15::e
2510:a1:12::d
2510:a1:24::}
92.7.11.1
92.7.7.6
2510:a1:3::{
2510:a1:6::f
2510:a1:2::X
92.7.14.3
92.7.28.0
92.7.5.1
92.7.8.3
2510:a1:f::d
2510:a1:4::e
2510:a1:1a::b
2510:a1:20::d
92.7.24.4
2510:a1:c::d
92.7.35.4
2510:a1:19::e
92.7.17.0
"""

print(f"Flag: {solve_abyss_final(logs)}")
```

- WE RUN THIS AND FINALLY GET OUR FLAG WHICH IS :: HQX{e1f63411d03d20dace954ebb0860d9f4}



```
PS D:\hackquest\Address Abyss> python .\clean_log.py
Flag: HQX{e1f63411d03d20dace954ebb0860d9f4}
```